Computers Math. Applic. Vol. 22, No. 2, pp. 39–47, 1991
Printed in Great Britain.

0097-4943/91 $3.00 + 0.00

PROPERTIES OF A CLASS OF TRIVALENT NETWORK GRAPHS AND OPTIMAL ROUTING

PRADIP K. SRIMANI
Department of Computer Science, Colorado State University
Ft. Collins, CO 80523
BHABANI P. SINHA AND BHARGAB B. BHATTACHARYA
Indian Statistical Institute, Calcutta, India
SURANJAN GHOSH
Jadavpur University, Calcutta, India

(Received July 1990 and in revised form December 1990)

Abstract—In the present paper we investigate various structural properties of a class of trivalent network graphs, known as Moebius graphs, and then propose an $O(N)$ optimal routing algorithm where N is the number of nodes in the graph. We also discuss the fault-tolerance aspects of such families of graphs.

1. INTRODUCTION

The interconnection topology plays an important role in the study of multiprocessor network design and modeling the network by an undirected graph consitutes a major part of interconnection topology research. In this model the vertices (nodes) of the graph denote the different processors or sites of the network, while the edges represent the full duplex communication lines between the nodes. An important assumption is that the network graph is connected and any node can reach any other node either directly (if they are adjacent in the graph) or via some intermediate nodes. Any interconnection structure used to realize a computer network should be able to accommodate a large number of sites and should maintain low upper bound on internode distance (to minimize communication delay) and also low upper bound on the number of connections per processor (to minimize the cost of the network). Given a set of processors the objective of designing a network graph is twofold: to keep the degree of each node minimum and to make the diameter of the graph as small as possible. These two objectives inherently conflict with each other and different authors have proposed and investigated quite a large number of graph structures. One of the most interesting family of such graphs is that of the trivalent network graphs [1,2], where each node is constrained to have at most three neighbors. The best known trivalent graphs [1], named as Moebius graphs, allow $N = 2^n$ processors to be connected in a network of diameter $\leq \lfloor 2n/2 \rfloor$ for any $n > 1$.

Authors in [1] have established the aforementioned lower bound on the diameter of the graph and have proposed an algorithm that computes a path between any pair of nodes of length equal to that bound. But this lower bound is not apparently achieved in a moderately sized Moebius graph, and more importantly, in any given Moebius graph minimal paths between many pairs of nodes are much smaller than the said bound. When a graph is to be used to construct computer networks, it is very essential to compute the *shortest* or the *minimal* path between arbitrary pair of nodes in order to minimize the inter-processor communication delay. Although the algorithm in [1] computes paths within the lower bound of the diameter of the graph, it almost never produces the minimal path between two given vertices. Our purpose in this paper is to investigate in details the algebraic structure of such Moebius graphs and then to utilize those properties to develop an efficient $O(N)$ algorithm for *optimal* routing between any pair of

vertices in the network graph. Throughout the rest of the paper we assume that the number N of nodes of the graph is 2^n for some integer $n \geq 1$.

2. PROPERTIES OF MOEBIUS GRAPHS

Let n be a fixed integer greater than 1 and let 2^n denote the set $S = \{s_0 s_1 \dots s_{n-1} | s_1 \in (0,1), 0 \leq i \leq n-1\}$.

DEFINITION. *The trivalent network graph, named as Moebius graph, or order n is defined to be an undirected graph $G = (V, E)$ with vertex set V, $|V| = 2^n$ and $(u, v) \in E$, iff either $u = f(v)$, or $f^{-1}(v)$, or $g(v)$, where f and g are two mapping functions $(2^n \to 2^n)$ given by:*

$$f(s_0 s_1 \dots s_{n-2} s_{n-1}) = s_1 s_2 \dots s_{n-1} \bar{s}_0$$
$$g(s_0 s_1 \dots s_{n-2} s_{n-1}) = s_0 s_1 \dots \bar{s}_{n-2} \bar{s}_{n-1}.$$

Here $(s_0 s_1 \dots s_{n-1})$ is the usual binary representation of any node in G. Similarly, the inverse functions f^{-1} and g^{-1} are defined as follows:

$$f^{-1}(s_0 \dots s_{n-2} s_{n-1}) = \bar{s}_{n-1} s_0 \dots s_{n-2},$$
$$g^{-1}(s_0 s_1 \dots s_{n-2} s_{n-1}) = s_0 s_1 \dots \bar{s}_{n-2} \bar{s}_{n-1}.$$

Obviously f is an *asymmetric* transition function (i.e., $f(u) = v$ does not, in general, imply $f(v) = u$), while g is *symmetric* ($g(u) = v$ always implies that $g(v) = u$), and, hence, the graph is *trivalent.* Moebius graphs of order 3 and 4 are shown in Figure 1.

LEMMA 1. *Let r and r' denote the run measures, i.e., the number of blocks of 1's and 0's in the binary representation of two nodes v and v' in G. If v' is reached by a single asymmetric transition, i.e., if $v' = f(v)$ or $v' = f^{-1}(v)$, then*

1) *$r' = r$ or $r - 1$, if r is even, and*
2) *$r' = r$ or $r + 1$, if r is odd.*

PROOF. Since f consists of taking out one bit from the left end of v and putting its complement at the right end of v in forming v', it is clear that r' can differ from r at most by 1. We also observe that, whenever a binary vector starts and ends with the same symbol, 0 or 1, r has to be odd, and if v starts and ends with different symbols, r has to be even. It then readily follows that if r is even and $v' = f(v)$, then r' can be either r or $r - 1$, and if r is odd, r' can be either r or $r + 1$. ∎

REMARK. It is clear that Lemma 1 holds when the function f is replaced by its inverse f^{-1}.

LEMMA 2. *Let r and r' denote the run measures of two nodes v and v'. If $v' = g(v)$, then $|r - r'| = 1$.*

PROOF. It follows directly from the definition of the function g that r' is either greater or less than r by exactly one. ∎

Again this lemma holds if g is replaced by g^{-1}.

THEOREM 1. *In a Moebius graph G, the run measures of any pair of adjacent vertices differ by at most 1.*

PROOF. It clearly follows from Lemma 1 and 2 and the definition of a Moebius graph.

DEFINITION. *Any cycle in a Moebius graph G consisting of only f-edges (induced by the asymmetric functions f or f^{-1}) is called an f-cycle. For example, {000, 001, 011, 111, 110, 100, 000} is an f-cycle, as shown in Figure 1a.*

DEFINITION. *The complement of a vertex u is a vertex v such that the binary representation of v is obtained by complementing the bits in the binary representation of u. For example, complement of the vertex 1011 is the vertex 0100 in the Moebius graph of order 4.*

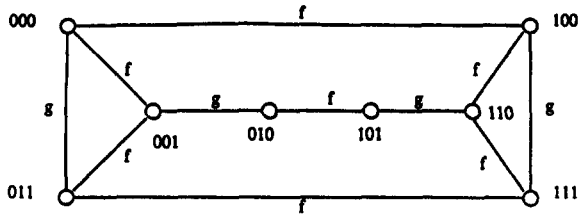

Figure 1a. Moebius graph for $n = 3$.

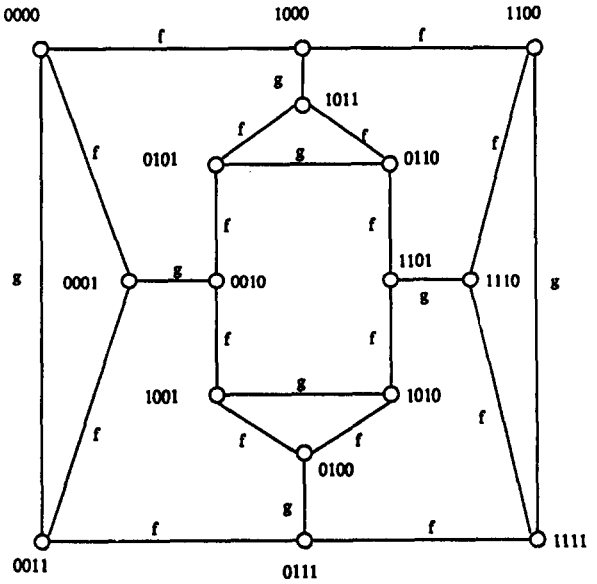

Figure 1b. Moebius graph for $n = 4$.

LEMMA 3. *All of the 2^n nodes of a Moebius graph of order n are partitioned into vertex disjoint f-cycles.*

PROOF. It follows directly from the definition that starting from an arbitrary vertex u, if the f function is repeatedly applied, a cycle of length at most $2n$ is traced in a Moebius graph of order n. Hence, all vertices are on some f-cycle. That these f-cycles are vertex disjoint follows from the observation that $f(v_1) = u = f(v_2)$, if and only if $v_1 = v_2$. ∎

LEMMA 4. *Any node and its complement must be in the same f-cycle.*

PROOF. It follows directly from the definition of the function f since any vertex can reach its complement by at most n repeated applications of the f-transition in a Moebius graph of order n. ∎

THEOREM 2. *In a Moebius graph of order n there is at least one f-cycle of length $2p$ iff $n = pq$ and q is odd.*

PROOF. Let p be a positive integer, $p \leq n$. Applying the f-transitions p times on any vertex $v_1 = s_0\, s_1 \ldots s_{n-2}\, s_{n-1}$, we can reach a vertex $v_2 = s_p\, s_{p+1} \ldots s_{n-1}\, \bar{s}_0\, \bar{s}_1 \ldots \bar{s}_{p-1}$. For v_2 to be the complement of v_1, a set of relations among the different bits of v_1 must hold which is termed the set of consistency relations induced by p. Let this be denoted by CR_p. Also let $n = p + k$. To derive the consistency relations, the following two cases are considered:

Case 1: $p \leq k$. In order that v_2 is complement of v_1, v_1 must be bit wise equal to $\bar{v}_2$.

$$v_1 = s_0\, s_1 \ldots s_{k-1}\, s_k\, s_{k+1} \ldots s_{n-1},$$
$$\bar{v}_2 = \bar{s}_p\, \bar{s}_{p+1} \ldots \bar{s}_{n-1}\, s_0\, s_1 \ldots s_{p-1}.$$

The bits on the right hand side of the dotted line are related by the following set of equations:

$$\begin{aligned} s_0 &= s_k, \\ s_1 &= s_{k+1}, \\ &\vdots \\ s_{p-1} &= s_{n-1}. \end{aligned} \tag{1}$$

Similarly the bits on the left hand side are related by the set of equations

$$\begin{aligned} s_p &= \bar{s}_0, \\ s_{p+1} &= \bar{s}_1, \\ &\vdots \\ s_{n-1} &= \bar{s}_{k-1}. \end{aligned} \tag{2}$$

It may be noted that the equations in (1) are of the form $s_i = s_{i+k}$, $0 \leq i \leq p-1$ and those in (2) are of the form $s_i = \bar{s}_{i-p}$, $p \leq i \leq n-1$. Hence it can be easily checked that the consistency relation involving s_0 will be of the form

$$s_0 = s_k = \bar{s}_{n-2p} = \cdots = s_0,$$

where the bits involved are always of the form $s_{\alpha n-\beta p}$ or $\bar{s}_{\alpha n-\beta p}$, $\alpha = \lceil \beta p/n \rceil$, β increases from 1, by a step of unity. This consistency relation is closed when $\alpha n = \beta p$, i.e., $\beta = n/(p,n)$, (p,n) being the GCD of p and n, and $\alpha = p/(p,n)$. As long as any equation from (1) is used to get the consistency relation, the value of $(\beta - \alpha)$ remains unchanged, as $k = n - p$. But the use of any equation from (2) increases the value of $(\beta - \alpha)$ by one. Furthermore, since $p \leq k$, to get the consistency relation, at least one equation from (2) is used between two successive uses of equations from (1). Hence the bits $s_{\alpha n-\beta p}$ appears in complimented form in the consistency relation if $(\beta - \alpha)$ is odd and in uncomplemented form otherwise. [For example, if $n = 22$, $p = 2$ and $k = 20$, the consistency relation for s_0 is given by: $s_0 = \bar{s}_{22-2.2} = s_{22-3.2} = \cdots = \bar{s}_{22-10.2} = s_{22-11.2}$. Again if $n = 22$, $p = 6$ and $k = 16$, the consistency relation for s_0 is given by $s_0 = s_{22-6} = \bar{s}_{22-2.6} = s_{22-3.6} = s_{2.22-4.6} = \bar{s}_{2.22-5.6} = s_{2.22-6.6} = \bar{s}_{2.22-7.6} = \bar{s}_{3.22-8.6} = s_{3.22-9.6} = \bar{s}_{3.22-10.6} = s_{3.22-11.6}$.]

Similarly the consistency relation for any bit s_i, $1 \leq i \leq p-1$, which is not included in the consistency relation of s_0, will include the bits of the form $s_{\alpha n-\beta p+i}$ or $\bar{s}_{\alpha n-\beta p+i}$ where, as before, $\alpha = \lceil \beta p/n \rceil$ and the bit will appear in complemented or uncomplemented form depending on whether $(\beta - \alpha)$ is odd or even. The number of bits in each of these consistency relations is $n/(p,n)$ since β increases from 1 to $n/(p,n)$ in steps of unity.

Case 2: $p > k$. In this case also the bits $s_0, s_1, \ldots, s_{n-1}$ are connected by Equations (1) and (2). The only difference is that between two successive uses of equations from (2) at least one equation from (1) must be used to get a consistency relation. It can then be easily checked as in Case 1, that the consistency relation for any bit s_i, $0 \leq i \leq p-1$, is of the form $s_i = s_{\alpha n-\beta p+i}$ or $\bar{s}_{\alpha n-\beta p+i}$ where $\alpha = \beta - \lfloor \beta k/n \rfloor = \lceil \beta p/n \rceil$, $1 \leq \beta \leq n/(p,n)$, and the bit is complemented or uncomplemented, depending on whether $(\beta - \alpha)$ is odd or even. [For example, if $n = 22$, $p = 18$ and $k = 4$, the consistency relation for s_0 is given by $s_0 = s_{22-18} = s_{2.22-2.18} = s_{3.22-3.18} = s_{4.22-4.18} = s_{5.22-5.18} = s_{5.22-6.18} = \bar{s}_{6.22-7.18} = \bar{s}_{7.22-8.18} = \bar{s}_{8.22-9.18} = \bar{s}_{9.22-10.18} = s_{9.22-11.18}$.]

Combining both the cases, it can be said that the consistency relation can be obtained, if and only if

$$(\beta - \alpha)_{\max} = \frac{n-p}{(p,n)} \quad \text{is even.}$$

We have to consider three possible cases:

Case a: Let $(p,n) = p$. Hence, $n = pq$, $q \geq 1$. Now for a consistency relation to exist, q must be odd.

Case b: Let μ be an integer such that $(\mu, n) = p$. Hence $\mu = ap$ where $a > 1$. The consistency relations for using f-transitions μ times can be obtained iff $n - \mu/(\mu, n)$ is even, i.e., a is odd. Next it is shown that the consistency relations obtained by using μ f-transitions are exactly the same as those with p f-transitions.

Let the bits in the consistency relations for μ f-transitions be of the form $s_{\sigma n-\delta\mu+i}$ or $\bar{s}_{\sigma n-\delta\mu+i}$, $0 \leq i \leq \mu - 1$, $1 \leq \delta \leq n/(\mu, n) = n/p = q$, $1 \leq \sigma \leq \mu/(\mu, n) = a$. Also let $\delta\mu = bn + r$, $0 \leq r < n = pq$. Hence $r = \delta\mu - bn = ap\delta - bpq$. Or we can write

$$r = p(a\,\delta \bmod q). \tag{3}$$

Similarly, $\sigma n - \delta\mu = n\lceil \delta\mu/n \rceil - \delta\mu = n(b+1) - bn - r = n - r$. Or we can write

$$\sigma n - \delta\mu = n - p(a\,\delta \bmod q) \text{ or } 0. \tag{4}$$

Since $(\mu, n) = p$ and $n = pq, \mu = ap$, we have $(a, q) = 1$. Let δ_1 and δ_2 be two distinct values of δ and let $a\delta_1 = m_1 q + r$ and $a\delta_2 = m_2 q + r$. So $a(\delta_1 - \delta_2) = (m_1 - m_2)\,q \neq 0$. Since $(a, q) = 1, (\delta_1 - \delta_2)$ must have q as a factor, which is impossible since $1 \leq \delta_1, \delta_2 \leq q$. Hence, $a\delta_1 \bmod q \neq a\delta_2 \bmod q$ when $\delta_1 \neq \delta_2$. Thus, from Equation (4), it can be seen that the bits $s_{\sigma n-\delta\mu+i}$ in the consistency relations for μ f-transitions are the same as those for p f-transitions. Furthermore, $a\delta = bq + c$, where $c = a\delta \bmod q$. So $a(\delta - \sigma) = bq - a\sigma + c = (\sigma - 1)\,q - a\sigma + c = \sigma(q - a) - q + c$. Hence, c must be odd when $(\delta - \sigma)$ is even and vice versa. In other words, $(\delta - \sigma)$ is even or odd, depending on whether $(c - 1)$ is even or odd. Since $(c - 1)$ is the value for $(\beta - \alpha)$ corresponding to a bit $s_{\alpha n-\beta p+1}$ in the consistency relation for p f-transitions, both p and μ produce the same consistency relation.

Case c: For any $\gamma < p$, $(\gamma, n) < p$. Hence the consistency relation for γ f-transitions will involve bits $s_{\lambda n-\xi p+i}$ or $\bar{s}_{\lambda n-\xi p+i}$, $1 \leq \xi \leq n/(\gamma, n)$. Hence, $\xi_{\max} > nq/p$, i.e., γ will lead to different consistency relations from those of p. It follows from (a), (b) and (c) that there is a cycle of half length p when $n = pq$ and q is odd. ∎

COROLLARY 1. *For all n there exists in a Moebius graph of order n at least one f-cycle of length $2n$.*

COROLLARY 2. *For odd n there exists exactly one f-cycle of length 2. For even n, no such f-cycle of length 2 exists.*

COROLLARY 3. *If n is prime, all but one f-cycles are of length $2n$. If n is a power of 2 all f-cycles are of length $2n$.*

COROLLARY 4. *Any f-cycle will contain vertices having at most two values of run measures, r and $r + 1$, where r is odd and $1 \leq r \leq n$.*

THEOREM 3. *If $(v_1, v_2, \ldots, v_{2p})$ denotes an f-cycle of a Moebius graph of order n and if $p < n$, then there does not exist any i, $j \leq 2p$, such that v_i is reached from v_j by a single symmetric transition.*

PROOF. We prove this by contradiction. Let $v = s_0\, s_1 \ldots s_{n-1}$ be a vertex in a f-cycle of length $2p$ where $n = pq$ and $q(>1)$ is odd. Since the complement of v is reached by applying the f-transition p times, it is clear that, for $0 \leq i < p$,

$$s_i = \bar{s}_{p+i} = s_{2p+i} = \bar{s}_{3p+i} = \cdots = s_{(q-1)p+i}. \tag{5}$$

Let $f^m(v)$ denote the vertex that is reached from v by successively applying f-transitions m times, $m < 2p$. Also let the same vertex be reached from v by a single g-transition, i.e., $f^m(v) = g(v)$. Now

$$f^m(v) = s_m\, s_{m+1} \ldots s_{n-1}\, \bar{s}_0\, \bar{s}_1 \ldots \bar{s}_{m-1}, \text{ and}$$
$$g(v) = s_0\, s_1 \ldots s_{n-3}\, \bar{s}_{n-2}\, \bar{s}_{n-1}.$$

Since $f^m(v)$ is assumed to be equal to $g(v)$, the following sets of equations are obtained:

$$\begin{aligned} s_0 &= s_m, \\ s_1 &= s_{m+1}, \\ &\vdots \\ s_{n-m-1} &= s_{n-1}, \end{aligned} \tag{6}$$

$$\begin{aligned} s_{n-m} &= \bar{s}_0, \\ s_{n-m+1} &= \bar{s}_1, \\ &\vdots \\ s_{n-3} &= \bar{s}_{m-3}, \end{aligned} \tag{7}$$

$$\begin{aligned} s_{n-2} &= s_{m-2}, \\ s_{n-1} &= s_{m-1}. \end{aligned} \tag{8}$$

Three possible range of values for m are considered now.

Case 1: $m = p$. This directly leads to contradiction, since by Equation (5) $s_0 = \bar{s}_p$, while from (6), $s_0 = s_m = s_p$.

Case 2: $m < p$. By Equation (8), $s_{n-1} = s_{m-1}$. Repeatedly applying Equations (6) and (5) and noting that with every use of (5) the bit gets complemented [this happens $m/(p,m) - 1$ times], we get that s_{n-1} is either s_{p-1} or $\bar{s}_{p-1}$. Similarly, by Equation (7) and repeated use of (5) and (6), we also get s_{n-3} is either $\bar{s}_{p-3}$ or s_{p-3}. Thus either of these two sets of relations $\{s_{n-1} = s_{p-1}$ and $s_{n-3} = \bar{s}_{p-3}\}$ and $\{s_{n-1} = \bar{s}_{p-1}$ and $s_{n-3} = s_{p-3}\}$ will lead to contradiction, as by Equation (5), $s_{n-1} = s_{p-1}$ and $s_{n-3} = s_{p-3}$, since $n = pq$ and q is odd.

Case 3: $p < m < 2p$. All of the above discussions are also valid for $m = p + r$, $0 \leq r < m$.

Thus for $p < n, g(v)$ cannot be equal to $f^m(v)$ for $m < 2p$. Hence, the theorem is proved. ∎

REMARK. It may be noted that for $p = n$, two vertices in an f-cycle may be linked by a single g-transition. This can be seen for $n = 3$ as in Figure 1. The next theorem deals with the vertex connectivity of Moebius graphs.

THEOREM 4. *For any two vertices u and v in a Moebius graph G, there exist at least two vertex disjoint paths connecting u and v.*

PROOF. It is noted that G is partitioned into vertex disjoint f-cycles and G remains connected via the symmetric transition g. Let us consider two such f-cycles c_1 and c_2 and let u and v be two vertices such that $u \in c_1$, $v \in c_2$ and $u = g(v)$. Also let $\bar{u}$ and $\bar{v}$ denote the complementary vertices of u and v, respectively. Then from earlier results, it follows that $g(\bar{v}) = \bar{u}$ and $\bar{u} \in c_1$ and $\bar{v} \in c_2$. Obviously, then the claim follows for all the vertices belonging to the two f-cycles c_1 and c_2. If the two cycles c_1 and c_2 are not directly connected by any g-transition they must be connected via other f-cycles and two vertex disjoint paths can be found accordingly. The main point to observe is that if two f-cycles are connected by a g-transition, there must be another g-transition between the complementary vertices, and any vertex and its complimentary vertex always belong to the same f-cycle. Moreover, if two vertices belong to the same f-cycle, the case is trivial, thereby proving the theorem. ∎

Theorem 4 has an important consequence which can be utilized to ensure correct communication of messages in presence of faulty processor nodes in the computer network. Since there exist at least two vertex-disjoint paths between any pair of vertices, if one of these paths is rendered ineffective due to faults developed in any one of the processors in that path, the other path can be invoked to ensure correct propagation of data once the presence of the fault is made known.

3. OPTIMAL ROUTING USING HEURISTIC SEARCH

In this section we utilize the general theory of heuristic search [3–5] to design an efficient $O(N)$ algorithm to compute the minimal path between any two given vertices in a Moebius graph with N vertices. Of the two given nodes (between which the path is to be computed) one is called the *start* node, denoted by s, and the other is called the *destination* or the goal node, denoted by d. The basic idea of using the heuristic search of computing a path from a start node to a destination node in a given symmetric graph is to generate all possible immediate successors of s and choose one among them which has the most potential to be its successor in the desired minimal path. The potential is measured in terms of an evaluation function that depends on the knowledge gained so far and some intrinsic heuristic estimate assigned to each node of the graph. This is repeated until the goal node is reached. The optimality of the computed path and the complexity of the algorithm depends on the choice of the evaluation function as well as the intrinsic heuristic estimates of the nodes.

In our case of a Moebius graph with $N(2^n)$ nodes, the cost of a path is defined to be the number of edges in the path, i.e., all the edges in the graph are assumed to have unit cost (any node can reach its adjacent node in constant time). We'll use the algorithm C of [4] to design our routing algorithm. We use the following evaluation function $\hat{z}(v)$ for any node v in G to direct the search:

$$\hat{z}(v) = \hat{y}(v) + \hat{e}(v),$$

where $\hat{y}(v)$ is the cost of the best path so far known, from the start node s to the node v, and $\hat{e}(v)$ represents the heuristic estimate assigned to this node, v. The choice of this heuristic estimate function for different nodes in the search graph will critically determine the search results. We propose that given the destination node d, any other node v in G is assigned a heuristic estimate $\hat{e}$ as:

$$\hat{e}(v) = |r(v) - r(d)|,$$

where the function r returns the run measure of the vertex. The routing algorithm can now be described as follows, which will compute the minimal path between any two given nodes s and d in a given Moebius graph G. Here **OPEN** and **CLOSED** are two simple lists which are initially empty and R is a real variable. By convention the start as well as the destination node are assigned zero heuristic estimate, i.e., $\hat{e}(s) = \hat{e}(d) = 0$.

The Routing Algorithm

Input: The graph G, two nodes s and d, and the the heuristic estimate function $\hat{e}$.
Output: Length of the minimal path between s and d in G.

Step 1: **OPEN** $:= s$; $R := 0$; $\hat{y}(s) := 0$; $\hat{z}(s) := 0$;

Step 2: Select a set S of nodes from **OPEN** such that $s = \{v | v \in$ **OPEN** and $\hat{z}(v) \leq R\}$. Select a node m from S such that $\hat{y}(m)$ is the smallest. If s is empty, then consider those nodes in OPEN with minimum $\hat{z}$ values and select from them a node m with minimum $\hat{y}$ value. Set $R := \hat{z}(m)$. [Ties are resolved arbitrarily but always in favor of goal nodes.] Remove m from **OPEN** and put it in **CLOSED**.

Step 3: If m happens to be the destination node d, then exit with $\hat{z}(m)$ as the output.

Step 4: Generate all the immediate successors of the node m (by applying the symmetric and asymmetric transition functions of the Moebius graph). For each immediate successor m' of m, set $y'(m') := \hat{y}(m) + 1$.

Step 5: If m' is not already in either **OPEN** or **CLOSED**, set $\hat{y}(m') := y'(m')$ and $\hat{z}(m') := \hat{y}(m') + \hat{e}(m')$. Put m' **in OPEN**.

Step 6: If m' is already in **OPEN** or in **CLOSED**, and $\hat{y}(m')$, then set $\hat{y}(m'): = y'(m')$ and $\hat{z}(m') = \hat{y}(m') + \hat{e}(m')$. If m' **is in CLOSED**, remove it and put it in **OPEN**.

Step 7: Go to Step 2.

REMARKS. The above routing algorithm outputs the length of the minimal path between two given nodes s and d. And it is easy to trace the minimal path by using an additional linked list to remember the nodes m selected at step 2 of the algorithm.

DEFINITION. *A heuristic estimate $\hat{e}$ is called admissible [4] if for all nodes $v \in G$, $\hat{e}(v) \leq e(v)$, where $e(v)$ denotes the actual length of the minimal path from v to d.*

THEOREM 5. *The heuristic estimate function $\hat{e}$ as proposed in this section is admissible.*

PROOF. We show that for any node v in the graph G, $\hat{e}(v) \leq e(v)$, where $e(v)$ denotes the minimal distance of v from the destination vertex d. Using Theorem 2, the result follows. ∎

THEOREM 6. *The proposed routing algorithm always computes the minimal path between any pair of start and destination vertices in the graph G.*

PROOF. Follows from Theorem 3.1 of [4], since under the proposed heuristic estimate the search is always admissible. ∎

DEFINITION. *Given a search graph G and a destination of node d, a heuristic estimate $\hat{e}$ is called consistent [3] iff for every pair of nodes (m, n) in G the following condition holds:*

$$\hat{e}(m) - \hat{e}(n) \leq c(m, n),$$

where $c(m, n)$ is the cost of the shortest path joining the nodes m and n.

THEOREM 7. *The heuristic estimate $\hat{e}(v)$ is consistent.*

PROOF. Assume the theorem is false. Then there exists a pair of nodes (m, n) such that

$$\hat{e}(m) - \hat{e}(n) > c(m, n).$$

Now we can write

$$\hat{e}(m) = |r(m) - r(d)|, \quad \hat{e}(n) = |r(n) - r(d)|.$$

Substitution in the earlier equation yields

$$|r(m) - r(d)| - |r(n) - r(d)| > c(m, n).$$

Since for any three integers A, B, and C we know that $|A - B| - |C - B| \leq |A - C|$, we get

$$|r(m) - r(n)| > c(m, n).$$

From Theorem 1 we know that the run measures of any two adjacent vertices in a Moebius graph differ by at most 1, and, hence, $c(m, n) \leq |r(m) - r(n)|$, which contradicts the above. Hence, the theorem. ∎

THEOREM 8. *The proposed routing algorithm has an worst-case execution time complexity of $O(N)$, where N is the number of nodes in the Moebius graph G.*

PROOF. Follows from the consistency of the heuristic estimate [3,4].

REMARKS. The proposed algorithm works even in the presence of faulty nodes. Theorem 4 tells us that in a Moebius graph G, for any two given vertices, there exist at least two vertex disjoint paths. In absence of any faulty node, our algorithm picks up the minimal path between the two nodes. Now if any node becomes faulty, that node should not be used in computing the path. This can be easily accomplished by checking the presence of any known faulty node in the list OPEN at every stage of node selection from OPEN in the Step 2 of the algorithm, and deleting that node from OPEN (without putting it in CLOSED). But if the faulty node happens to invalidate the only minimal path between the nodes s and d, our algorithn will pick up the next best path (path of next smallest length).

4. CONCLUSION

In this paper, we have investigated, in detail, different structural properties of Moebius graphs which can be used for implementing computer networks. Then we have used those properties to design an efficient routing algorithm that computes the shortest path between any two given nodes in the network graph in only $O(N)$ time. This is a significant improvement over the existing algorithm in [1], which computes a path always equal to the upper bound on diameter of the graph. We have shown that this family of Moebius graphs has also the fault-tolerance properties, i.e., for any two given nodes there exist in a Moebius graph two vertex disjoint paths. Thus any computer network developed on a Moebius graph is able to tolerate any arbitrary single node failure. We have also shown that our proposed routing algorithm works even in the presence of such node failures; it always picks up the shortest available path in the network, bypassing the faulty node.

REFERENCES

1. W.E. Leland and M.H. Solomon, Dense trivalent graphs for processor interconnection, *IEEE Trans. Comput.* C-31, 219–222 (March 1982).
2. B.W. Arden and H. Lee, A multi-tree structures network, *Proc. Compcon*, 201–210 (Fall 1978).
3. N.J. Nilsson, Principles of Artificial Intelligence, *Tioga* (1980).
4. A. Bagchi and A. Mahanti, Search algorithms under different kinds of heuristics: A comparative study, *Journal of ACM* 30, 1–21 (January 1983).
5. J. Pearl, *Heuristics*, Addison-Wesley, (1984).